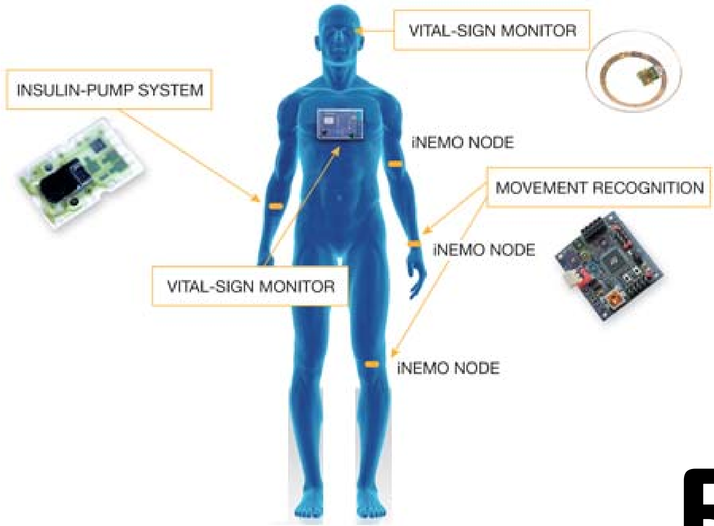
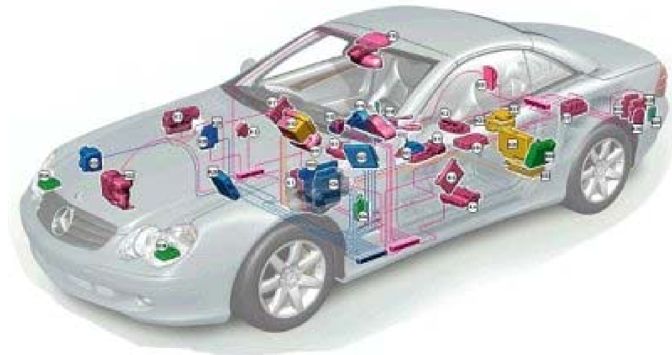

Timers
Part 1





Real Time
Embedded Systems

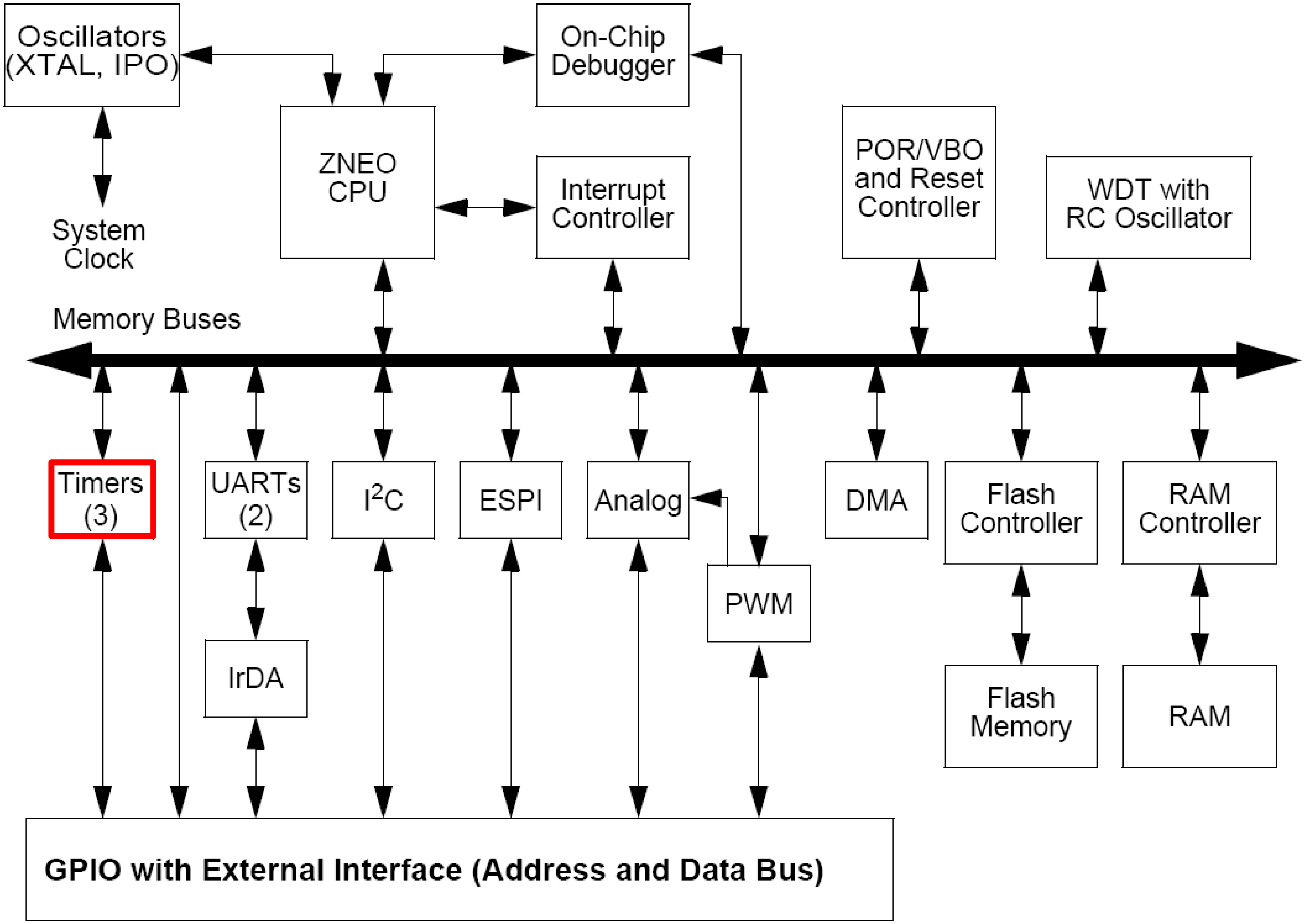
www.atomicrhubarb.com/embedded

Lecture 1 – January 17, 2012

Topic

Section Topic

- Where in the books
  - Catsoulis chapter/page
  - Simon chapter/page
  - Zilog UM197 (ZNEO Z16F Series Flash Microcontroller Contest Kit User Manual)
  - Zilog UM171 (ZiLOG Developer Studio II—ZNEO User Manual)
  - Zilog PS220 (ZNEO Z16F Series Product Specification)
  - Zilog UM188 (ZNEO CPU Core User Manual)
  - Assorted datasheets


Oscillators (XTAL, IPO)
On-Chip Debugger
ZNEO CPU
Interrupt Controller
POR/VBO and Reset Controller
WDT with RC Oscillator
System Clock
Memory Buses
Timers (3)
UARTs (2)
I2C
ESPI
Analog
DMA
Flash Controller
RAM Controller
PWM
IrDA
Flash Memory
RAM
GPIO with External Interface (Address and Data Bus)

What is a Timer

(a microcontroller timer)

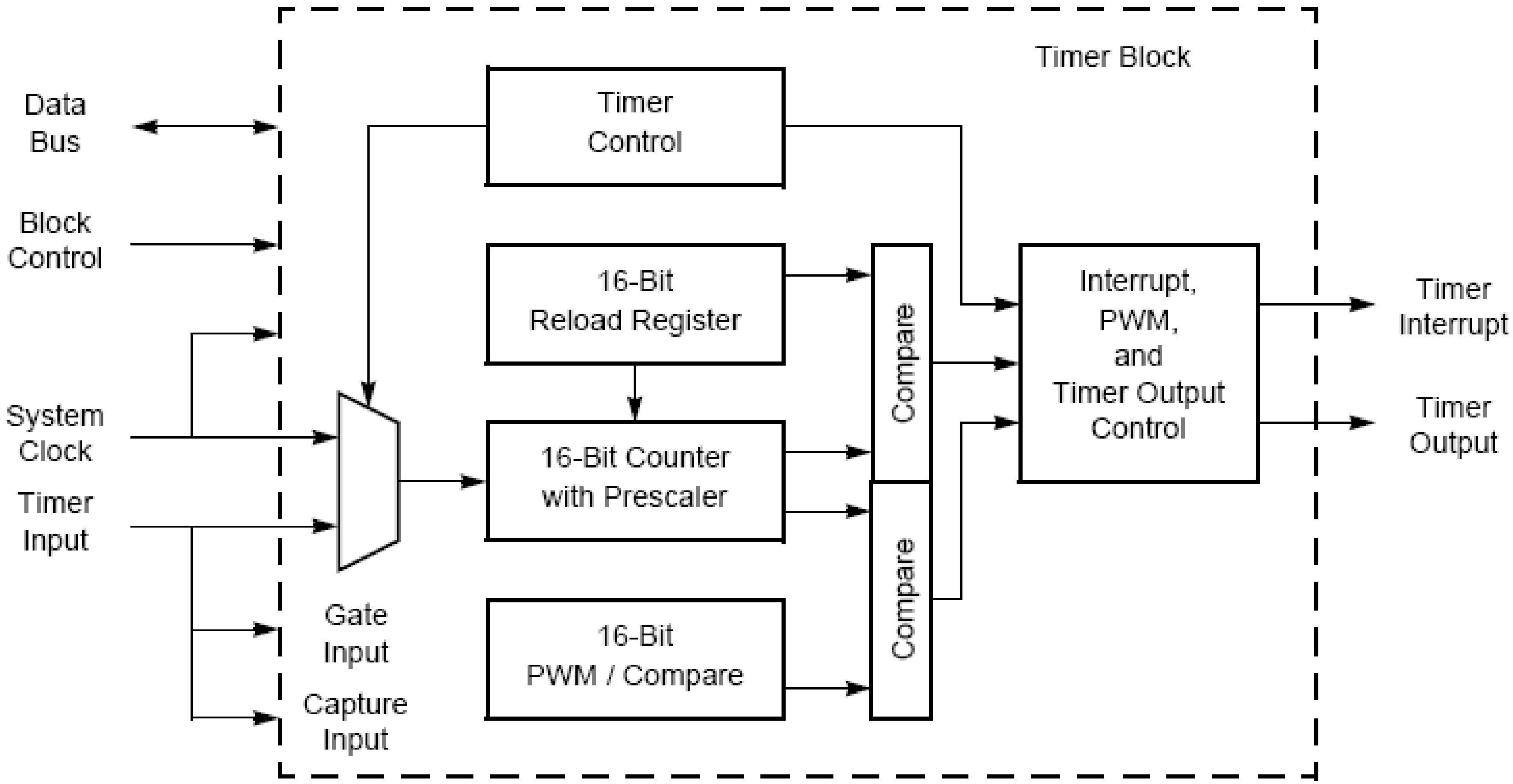
Timers

- Most microprocessors and microcontrollers have some sort of timers. Some have as many as sixteen timers.
- These are usually just <u>digital counters</u> that are set to a number by software, and then count down or down to zero.
- When they reach zero, they generate an interrupt or set a flag or something.

ZNEO Timers

- 16 bit counter that increments with system clock, or external signal.
- Interrupts can be triggered on counter reaching certain values.
- Other things can happen on counter reaching certain values (depends on the mode).
- 12 different modes

Timer Diagram



Timers

- 3 timers
- Timer0, Timer1, Timer2

Timer Modes

- One-Shot
- Continuous
- Counter
- PWM single output
- Capture
- Compare
- Gated
- Capture/Compare
- PWM dual output
- Capture Restart
- Comparator Counter
- Triggered One-Shot

Timers Have Registers

- Lots of registers …
- TxH, TxL - 16 bit timer value (current time)
- TxRH, TxRL - 16 bit reload value
- TxPWMH, TxPWML - 16 bit PWM value (and other uses)
- TxCTL0,1 = controls the timer configuration

x = 0,1,2

Timers - Base Address = FFF_E300				
Timer 0 (General-Purpose Timer) Base Address = FF_E300				
FF_E300	Timer 0 High Byte	T0H	00	105
FF_E301	Timer 0 Low Byte	T0L	01	105
FF_E302	Timer 0 Reload High Byte	T0RH	FF	106
FF_E303	Timer 0 Reload Low Byte	T0RL	FF	106
FF_E304	Timer 0 PWM High Byte	T0PWMH	00	106
FF_E305	Timer 0 PWM Low Byte	T0PWML	00	107
FF_E306	Timer 0 Control 0	T0CTL0	00	107
FF_E307	Timer 0 Control 1	T0CTL1	00	109

Timer 1 (General-Purpose Timer) Base Address = FF_E310				
FF_E310	Timer 1 High Byte	T1H	00	105
FF_E311	Timer 1 Low Byte	T1L	01	105
FF_E312	Timer 1 Reload High Byte	T1RH	FF	106
FF_E313	Timer 1 Reload Low Byte	T1RL	FF	106
FF_E314	Timer 1 PWM High Byte	T1PWMH	00	106
FF_E315	Timer 1 PWM Low Byte	T1PWML	00	107
FF_E316	Timer 1 Control 0	T1CTL0	00	107
FF_E317	Timer 1 Control 1	T1CTL1	00	109

Table 62. Timer 0-2 Control 0 Register (TxCTL0)

BITS	7	6	5	4	3	2	1	0
FIELD	TMODE[3]	TICONFIG		CASCADE	PWMD			INCAP
RESET	0	00		0	000			0
R/W	R/W	R/W		R/W	R/W			R
ADDR	FF-E306H, FF-E316H, FF-E326H							

Table 63. Timer 0-2 Control 1 Register (TxCTL1)

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	TPOL	PRES			TMODE		
RESET	0	0	000			000		
R/W	R/W	R/W	R/W			R/W		
ADDR	FF-E307H, FF-E317H, FF-E327H							

Bit Position	Value (H)	Description
[7] TEN		Timer is enabled.
	0	Timer is disabled.
	1	Timer is enabled.
[6] TPOL		**Timer Input/Output Polarity** This bit is a function of the current operating mode of the timer. It determines the polarity of the input and/or output signal. When the timer is disabled, the timer output signal is set to the value of this bit. **ONE-SHOT mode** —If the timer is enabled, the timer output signal pulses (changes state) for one system clock cycle after timer Reload. **CONTINUOUS mode**—If the timer is enabled, the timer output signal is complemented after timer Reload.

...

[5–3] PRES		The timer input clock is divided by 2^{PRES}, where PRES is set from 0 to 7. The prescaler is reset each time the timer is disabled. This ensures proper clock division each time the timer is restarted.
	000	Divide by 1
	001	Divide by 2
	010	Divide by 4
	011	Divide by 8
	100	Divide by 16
	101	Divide by 32
	110	Divide by 64
	111	Divide by 128
[2–0] TMODE[2:0]		This field along with the `TMODE[3]` bit in T0CTL0 register determines the operating mode of the timer. TMODE[3:0] selects from the following modes:
	0000	ONE-SHOT mode
	0001	CONTINUOUS mode
	0010	COUNTER mode
	0011	PWM SINGLE OUTPUT mode
	0100	CAPTURE mode
	0101	COMPARE mode
	0110	GATED mode
	0111	CAPTURE/COMPARE mode
	1000	PWM DUAL OUTPUT mode
	1001	CAPTURE RESTART mode
	1010	COMPARATOR COUNTER mode
	1011	TRIGGERED ONE-SHOT mode

Bit Position	Value (H)	Description
[7] TMODE[3]		**Timer Mode High Bit** This bit along with TMODE[2:0] field in T0CTL1 register determines the operating mode of the timer. This is the most significant bit of the timer mode selection value. For more details, see the T0CTL1 register description.
[6–5] TICONFIG		**Timer Interrupt Configuration—**This field configures timer interrupt definitions. These bits affect all modes. The effect per mode is explained below: ONE SHOT, CONTINUOUS, COUNTER, PWM, COMPARE, DUAL PWM, TRIGGERED ONE-SHOT, COMPARATOR COUNTER: 0x Timer interrupt occurs on reload. 10 Timer interrupts are disabled. 11 Timer Interrupt occurs on reload. **GATED**: 0x Timer interrupt occurs on reload. 10 Timer interrupt occurs on inactive gate edge. 11 Timer interrupt occurs on reload. **CAPTURE, CAPTURE/COMPARE, CAPTURE RESTART**: 0x Timer interrupt occurs on reload and capture. 10 Timer interrupt occurs on capture only. 11 Timer interrupt occurs on reload only.

[4] CASCADE	 0 1	**Timer cascade—**This field allows the timers to be cascaded for larger counts. The timer is not cascaded. Timer is cascaded. If timer 0 CASCADE bit is set, COMPARATOR output is used as input. If timer 1 CASCADE bit is set, the Timer 0 output is used as the input. If timer 2 CASCADE bit is set, the timer 1 output is used as input.
[3:1] PWMD	 000 001 010 011 100 101 110 111	**PWM Delay Value** This field is a programmable delay to control the number of additional system clock cycles following a PWM or Reload compare before the timer output or the timer output complement is switched to the active state. This field ensures a time gap between deassertion of one PWM output to the assertion of its complement. No delay. 2 cycles delay. 4 cycles delay. 8 cycles delay. 16 cycles delay. 32 cycles delay. 64 cycles delay. 128 cycles delay.
[0] INCAP	 0	**Input Capture Event** Previous timer interrupt is not a result of a timer input capture event.
	1	Previous timer interrupt is a result of a timer input capture event.

Timer Input/Output Pins

General-Purpose Timers		
T0OUT/$\overline{\text{T0OUT}}$ T1OUT/$\overline{\text{T1OUT}}$ T2OUT/$\overline{\text{T2OUT}}$	O	**General-purpose timer outputs**: These signals are output pins from the timers.
T0IN/T0IN1/T0IN2 /T1IN/T2IN	I	**General-purpose timer inputs**: These signals are used as the capture, gating, and counter inputs.

Timer IO pins

- TxOUT – Changes state on timeout
- /TxOUT – The inverse of TxOUT
- (TPOL register determines direction of state change)
- TxIN – Acts as an enable or clock source on some timer modes.

Timer Modes

- One-shot
- Triggered One-Shot
- Continuous
- Counter, Comparison Counter
- Pule Width Modulated
- Capture, Capture Restart
- Compare
- Gated
- Capture/Compare

One Shot Mode

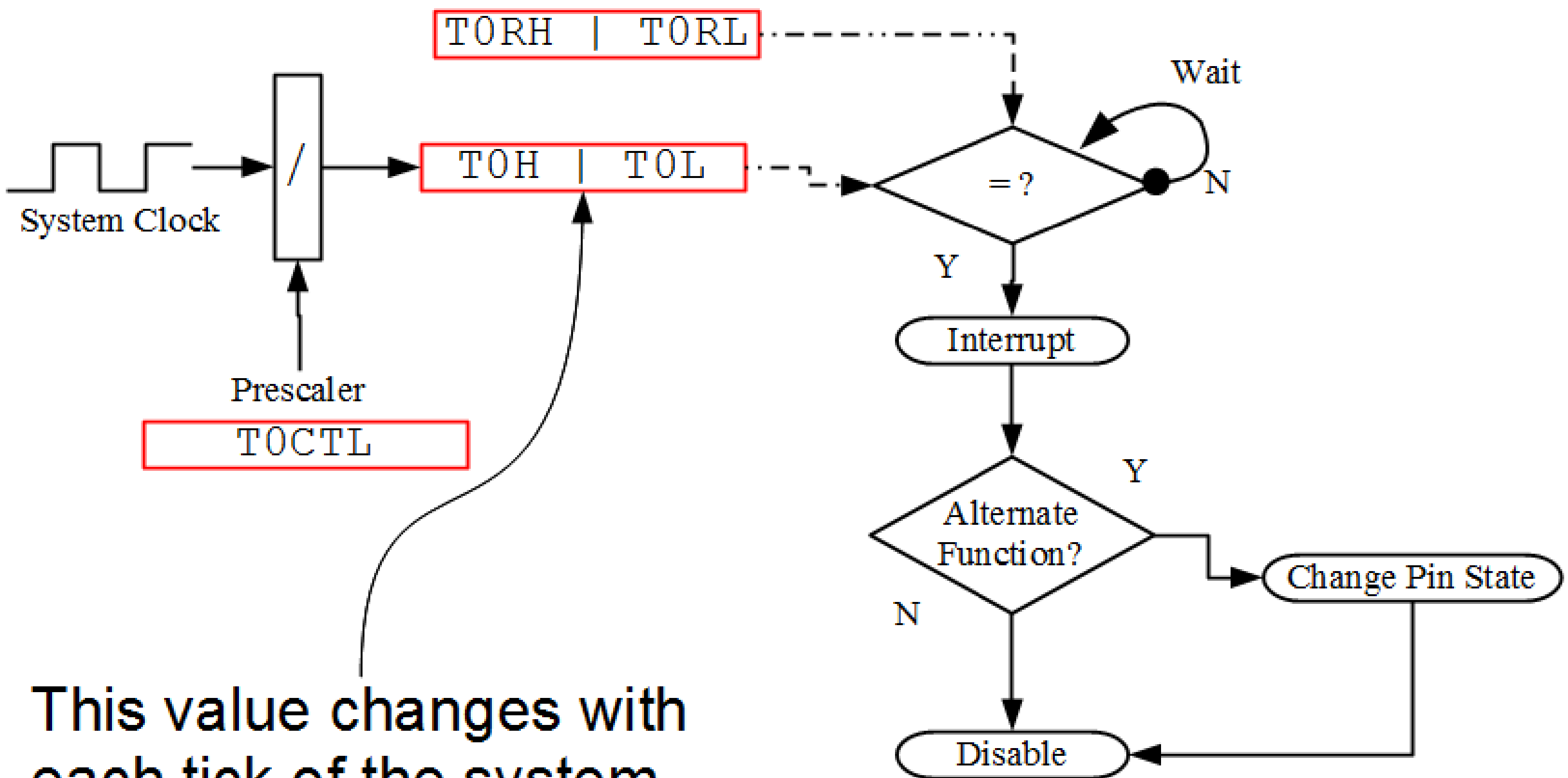
- Timer counts up to the 16-bit Reload value
- The timer input is the system clock.
- Upon reaching the Reload value,
  - generates an interrupt
  - count value in the Timer High/Low registers is reset to 0001H.
  - the timer is automatically disabled and stops counting.
- If Timer Output alternate function is enabled, the Timer Output pin changes state for one system clock cycle (from Low to High or from High to Low)

One Shot

- Timer Output can make a permanent state change upon One-Shot timeout by setting the TPOL bit in the Timer Control Register to the start value before beginning One-Shot mode. Then, after starting the timer, set TPOL to the opposite bit value.
- If interrupts are not enabled, no ISR is called but a bit in IRQ0,1,2 is still set to indicate timer timeout.

One-Shot

- Z16 Manual has very wordy description of each timer mode.
- I have a flow-like diagram which I think is clearer.
  - Not intended to be an accurate representation of hardware, but to show the registers and the sequence of events.
- <u>Read</u> the description in the product specification.



This value changes with each tick of the system clock

How Long?

The timer period is calculated by the following equation (start value = 1):

$$\textbf{One-Shot Mode Time-Out Period (s)} = \frac{(\textbf{Reload Value} - \textbf{Start Value} + \textbf{1}) \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

$$reload = ((timeout * clock) / prescale) + startvalue - 1$$

OneShot Timer
Recipe

One Shot

1. Write to Timer Control Register to set:
    a. disable timer
    b. configure for one shot
    c. set prescale value
    d. if using alternate function, set initial output level
2. Write Timer High and Timer Low byte registers
3. Write to Timer Reload High/Low registers
4. Enable timer interrupt if desired, and set timer interrupt priority by writing to the interrupt registers
5. If using Timer Output, configure the associated GPIO port pin for alternate function
6. Write to the Timer Control Register to enable timer

$$\text{One-Shot Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Time and Clocks

- Time is based on the system clock.
- What is the system clock (how fast?)?

Clocks

- The ZNEO has several
  - A 5.5 MHz internal clock
  - Supports external crystal (up to 20 MHz)
    - 18.432 MHz on board
  - Supports external RC oscillator
    - If we had one (we don't).
- You can switch them but for the time being, 5.5 MHz is the system clock.

One Shot Time

$$\text{One-Shot Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

- Max time.
  - Reload = FFFF
  - Prescale = 128
  - (65535 * 128)/5500000 = 1.525 seconds
- Minimum
  - Reload = 0001
  - Prescale = 1
  - (1 * 1)/5500000 = 0.182 us
    - 182 ns! or 1 system clock cycle

Time

- To get shorter times (which we will need) we need to switch to the 18.432 MHz clock.
- Z16F is specified to run with a 20 MHz clock maximum.
- Can overclock this some (24 MHz in some cases).

Why would you use a one-shot timer?

Triggered One-Shot Mode

- Just like the one-shot but requires an external pin to start the timer.
- TxIN is used to start the timer.
- Must configure the appropriate GPIO pin for TxIN (PA0=T0IN, PC0=T1IN, PC6=T2IN)
- The TPOL bit is used to configure the TxIN pin for rising edge (1) or falling edge (0).

Triggered OneShot Timer Recipe

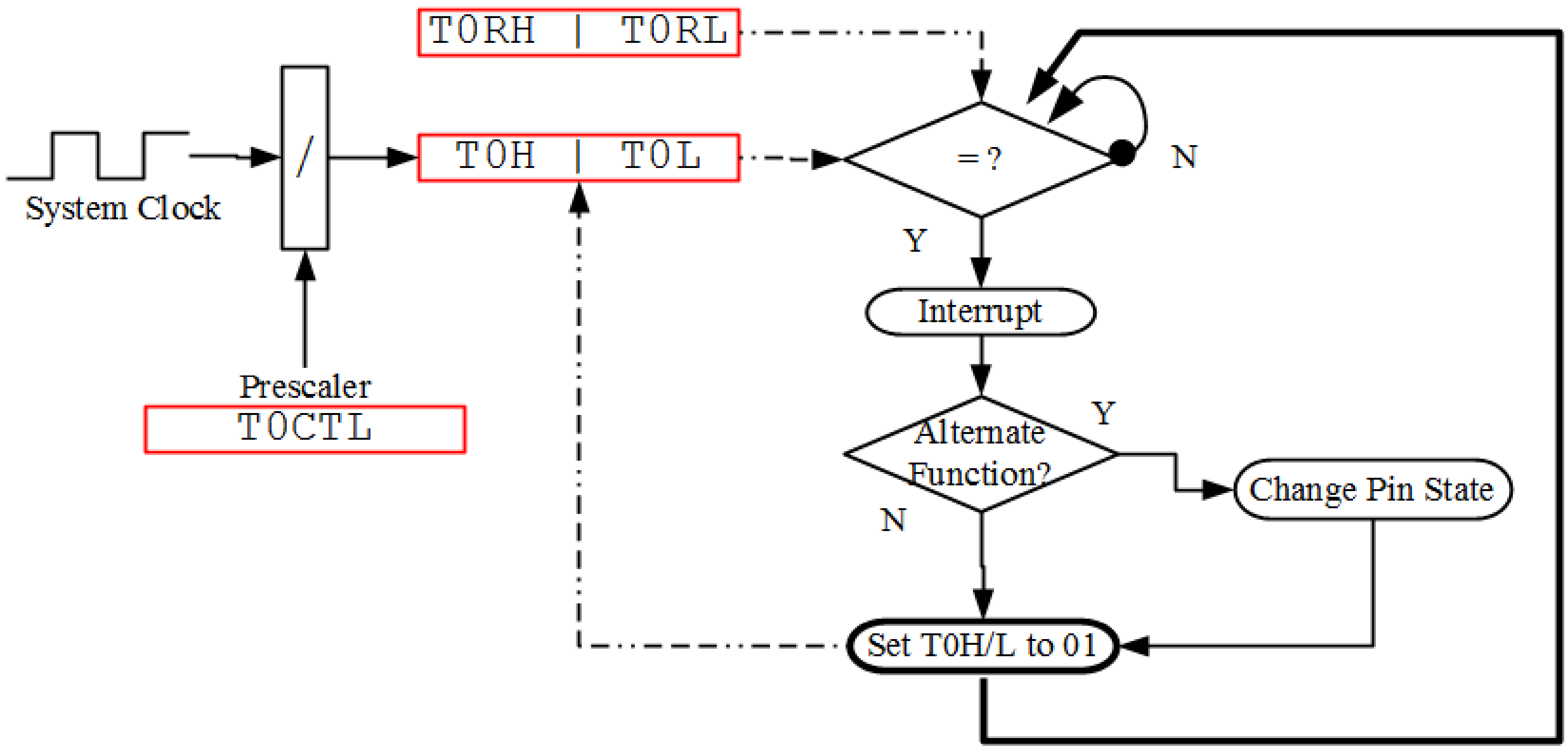
Triggered One Shot

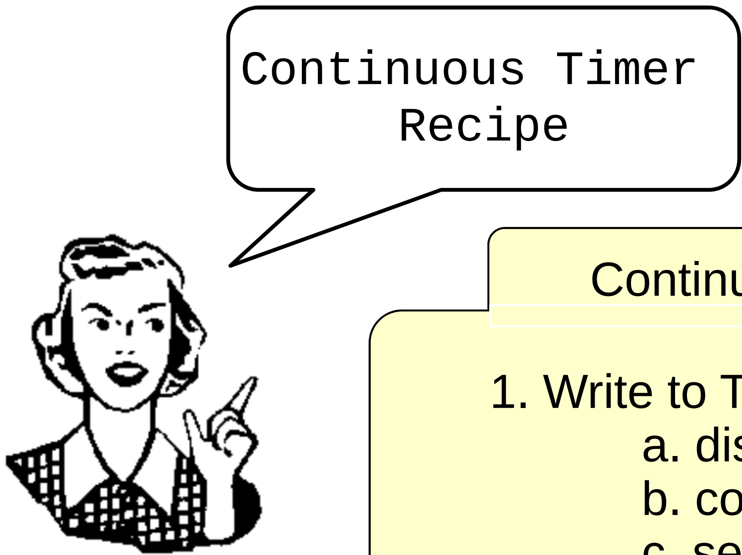
1. Write to Timer Control Register to set:
    a. disable timer
    b. configure for triggered one shot
    c. set prescale value
    d. set TPOL
2. Write Timer High and Timer Low byte registers
3. Write to Timer Reload High/Low registers
4. Enable timer interrupt if desired, and set timer interrupt priority by writing to the interrupt registers
5. Configure the associated GPIO port pin for alternate function for TxIN and TxOUT
6. Write to the Timer Control Register to enable timer.

$$\text{One-Shot Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Continuous Mode

- Just like the one shot timer except that when the timer value reaches the reload value, the timer starts over again at 1.
- Forever.
- Until disabled.


T0RH | T0RL
System Clock
/
T0H | T0L
=?
N
Y
Interrupt
Prescaler
T0CTL
Y
Alternate Function?
Change Pin State
N
Set T0H/L to 01



Continuous

1. Write to Timer Control Register to set:
   a. disable timer
   b. configure for continuous
   c. set prescale value
   d. if using alternate function, set initial output level
   and pin polarity
2. Write Timer High and Timer Low byte registers
3. Write to Timer Reload High/Low registers
4. Enable timer interrupt if desired, and set timer interrupts priority by writing to the interrupt registers
5. If using Timer Output, configure the associated GPIO port pin for alternate function
6. Write to the Timer Control Register to enable timer.

$$\text{Continuous Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Continuous Time

$$\text{Continuous Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

- One interrupt per timeout.
- One output pin state change per timeout.
- So the period of output = <u>2 times</u> the TimeOut (Frequency = 1/Period) !

This is where we Stopped last week

Why use a continuous timer?

Review what we started last week

Continuous

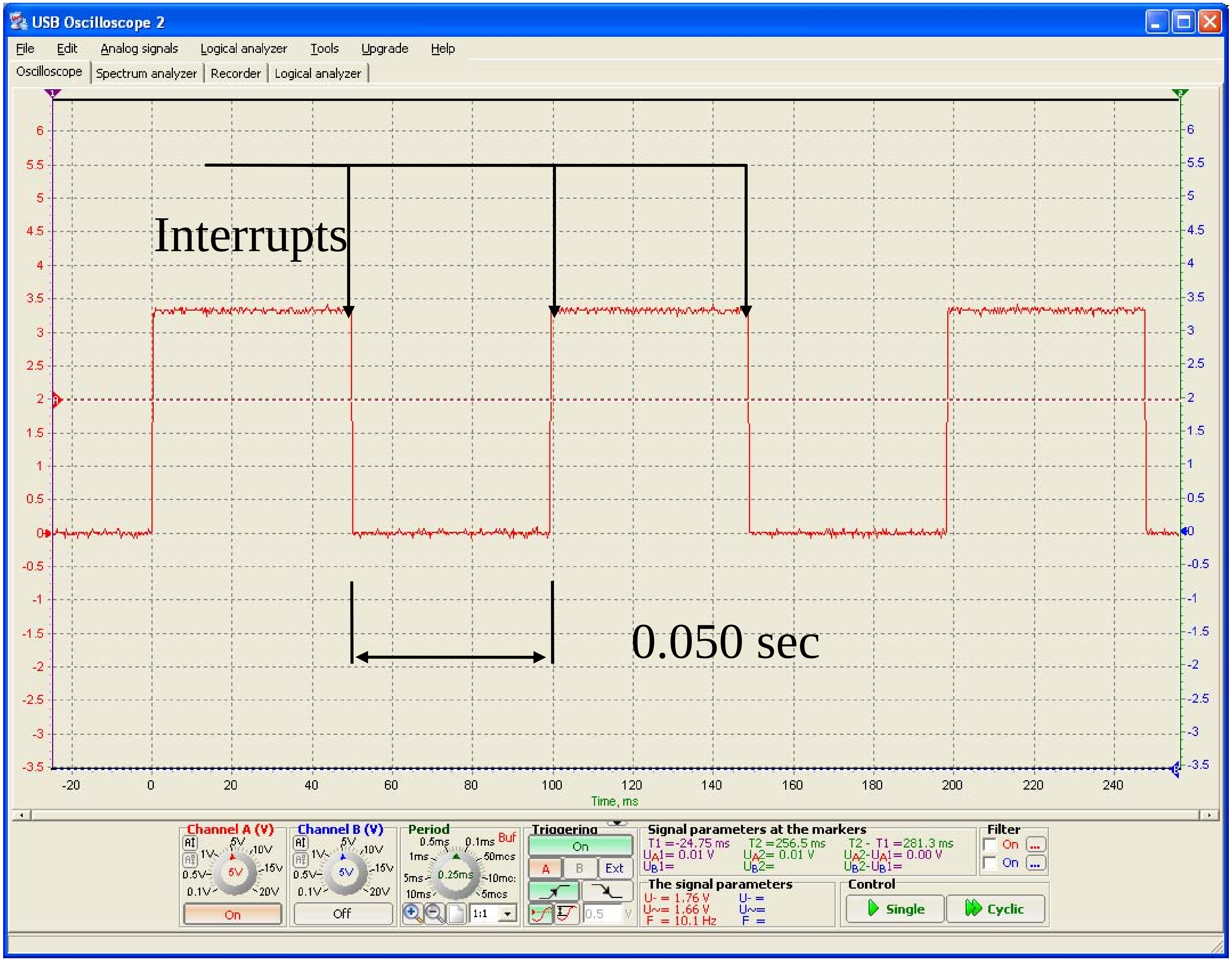
1. Write to Timer Control Register to set:
   a. disable timer
   b. configure for continuous mode
   c. set prescale value
   d. if using alternate function, set initial output level and pin polarity
2. Write Timer register (T0HL)
3. Write to Timer Reload High/Low registers (T0R)
4. Enable timer interrupt if desired, and set timer interrupts priority by writing to the interrupt registers
5. If using Timer Output, configure the associated GPIO port pin for alternate function
6. Write to the Timer Control Register to enable timer

$$\text{Continuous Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Continuous Example

Example
ContinuousTimer.c

- Example
  - Configure Timer0 for 50ms output
  - Enable T0OUT
- Use oscilloscope to examine Timer Output 0
- There are several ways to code the timer values.



USB Oscilloscope 2
File
Edit
Analog signals
Logical analyzer
Tools
Upgrade
Help
Oscilloscope
Spectrum analyzer
Recorder
Logical analyzer
Interrupts
0.050 sec
Time, ms
Channel A (V)
Channel B (V)
Period
Triggering
Signal parameters at the markers
T1 =-24.75 ms
T2 =256.5 ms
T2 - T1 =281.3 ms
The signal parameters
Control
Single
Cyclic
Filter

10 Hz?

- Recall that output pin changes with EVERY timeout.
- So output frequency is 1/(2 * timeout)
- Or 10 Hz not 20 Hz !

What's T0HL, T0R?

- T0HL = T0H, T0L
- T0R = T0RH, T0RL

- How do we know this?

Low level
SFR programming
<u>hex, binary magic numbers</u>

```
void init_timer_1(void) {

    T0CTL1 |= 0x7F;           // Disable timer (why?)

    T0CTL1 = 0x39;            // 0011_1001 0=disable
                              //   0=tpol, 111=prescale, 001=mode
    T0CTL0 = 0x00;            // 0=mode, 00=ticonfig, 0=cascade,
                              //   000=pwmd, 0=incap

    T0L = 0x01;               // Initial counter value
    T0H = 0x00;

    //   reload = clock / prescale * timeout
    //   desired timeout is 0.05 seconds
    //   reload = 5500000 / 128 * 0.05 = 0x0864
    T0R = 0x0864;

       IRQ0ENL |= 0x20;          // Enable Timer0 interrupt
       IRQ0ENH |= 0x20;

       // Configure the Output pin (so we can see it on scope)
       PADD &= ~0x02;
       PAAF |= 0x02;

       T0CTL1 |= 0x80;           // Enable Timer0
}
```

High level
SFR <u>byte</u> programming
using #define to make
<u>readable code</u>

```
// Timer enable and disable
#define TIMER_DISABLE 0x00
#define TIMER_ENABLE  0x80

// Timer modes
#define TIMER_MODE_ONESHOT        0x00
#define TIMER_MODE_CONTINUOUS     0x01
#define TIMER_MODE_COUNTER        0x02
#define TIMER_MODE_PWM            0x03
#define TIMER_MODE_CAPTURE        0x04
#define TIMER_MODE_COMPARE        0x05
#define TIMER_MODE_GATED          0x06
#define TIMER_MODE_CAPTURECOMPARE 0x07

// Timer prescale values
#define TIMER_PRESCALE_1   0x00
#define TIMER_PRESCALE_2   0x08
#define TIMER_PRESCALE_4   0x10
#define TIMER_PRESCALE_5   0x18
#define TIMER_PRESCALE_16  0x20
#define TIMER_PRESCALE_32  0x28
#define TIMER_PRESCALE_64  0x30
#define TIMER_PRESCALE_128 0x38

// Timer output pin polarity
#define TIMER_TPOL_1 0x40
#define TIMER_TPOL_0 0x00

// Timer Interupt Configuration
#define TIMER_TICONFIG_RELOAD   0x00
#define TIMER_TICONFIG_DISABLED 0x40
```

```
// Timer cascade
#define TIMER_CASCADE     0x10
#define TIMER_CASCADE_NOT 0x00

// Timer PWM delay
#define TIMER_PWM_DELAY_0  0x00
#define TIMER_PWM_DELAY_2  0x02
#define TIMER_PWM_DELAY_4  0x04
#define TIMER_PWM_DELAY_8  0x06
#define TIMER_PWM_DELAY_16 0x08

// Timer capture mode
#define TIMER_INPUT_CAPTURE_OFF 0x00
#define TIMER_INPUT_CAPTURE_ON  0x01

// These are IRQ0 register bits.
// All 3 IRQ registers or all 24 bits.
#define IRQ_Timer0 0x20
#define IRQ_Timer1 0x40
#define IRQ_Timer2 0x80

// Some binary
#define b00000000 0x00
#define b00000001 0x01
#define b00000010 0x02
...
#define b00001110 0x0E
#define b00001111 0x0F

#define CLOCK 5500000 // Clock in HZ
```

```
void init_timer_2(void) {

    T0CTL1 = TIMER_DISABLE;
    T0CTL1 = TIMER_MODE_CONTINUOUS + TIMER_PRESCALE_128 + TIMER_TPOL_0;

    // alternate form ...
    T0CTL1 =  TIMER_DISABLE | TIMER_MODE_CONTINUOUS
            | TIMER_PRESCALE_128 | TIMER_TPOL_0;

    T0CTL0 = TIMER_CASCADE_NOT + TIMER_PWM_DELAY_0 + TIMER_INPUT_CAPTURE_OFF;

    // Initial counter value
    T0HL = 0x00;

    // Timer reload
    //   reload = clock / prescale * timeout
    //   desired timeout = 0.05
    T0R = CLOCK / 128 * 0.05;

      // Enable Timer0 interrupt
    IRQ0EN = IRQ_Timer0;

      // Configure the Output pin
      PADD &= ~b00000010;
      PAAF |= b00000010;

      T0CTL1 |= TIMER_ENABLE;

}
```

```
void init_timer_2(void) {

    T0CTL1 = TIMER_DISABLE;

    T0CTL1 = TIMER_MODE_CONTINUOUS + TIMER_PRESCALE_128 + TIMER_TPOL_0;

    T0CTL0 = TIMER_CASCADE_NOT + TIMER_PWM_DELAY_0 + TIMER_INPUT_CAPTURE_OFF;

    // Initial counter value
    T0HL = 0x00;

    // Timer reload
    //   reload = clock / prescale * timeout
    //   desired timeout = 0.05
    T0R = CLOCK / 128 * 0.05;

       // Enable Timer0 interrupt
    IRQ0EN = IRQ_Timer0;

       // Configure the Output pin
       PADD &= ~b00000010;
       PAAF |= b00000010;

       T0CTL1 |= TIMER_ENABLE;

}
```

Notice that T0HL is 16-bit

So is T0R

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	TPOL	PRES			TMODE		
RESET	0	0	000			000		
R/W	R/W	R/W	R/W			R/W		
ADDR	FF-E307H, FF-E317H, FF-E327H							

[5–3]
PRES

The timer input clock is divided by 2^{PRES}, where
The prescaler is reset each time the timer is disak
clock division each time the timer is restarted.

000	Divide by 1
001	Divide by 2
010	Divide by 4
011	Divide by 8
100	Divide by 16
101	Divide by 32
110	Divide by 64
111	Divide by 128

```
// Timer prescale values
#define TIMER_PRESCALE_1   0x00
#define TIMER_PRESCALE_2   0x08
#define TIMER_PRESCALE_4   0x10
#define TIMER_PRESCALE_8   0x18
#define TIMER_PRESCALE_16  0x20
#define TIMER_PRESCALE_32  0x28
#define TIMER_PRESCALE_64  0x30
#define TIMER_PRESCALE_128 0x38
```

```
// Timer prescale values
#define TIMER_PRESCALE_1   0x00
#define TIMER_PRESCALE_2   0x01
#define TIMER_PRESCALE_4   0x02
#define TIMER_PRESCALE_8   0x03
#define TIMER_PRESCALE_16  0x04
#define TIMER_PRESCALE_32  0x05
#define TIMER_PRESCALE_64  0x06
#define TIMER_PRESCALE_128 0x07
```

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	TPOL	PRES			TMODE		
RESET	0	0	000			000		
R/W	R/W	R/W	R/W			R/W		
ADDR	FF-E307H, FF-E317H, FF-E327H							

```
// Timer prescale values
#define TIMER_PRESCALE_1   0x00
#define TIMER_PRESCALE_2   0x08
#define TIMER_PRESCALE_4   0x10
#define TIMER_PRESCALE_8   0x18
#define TIMER_PRESCALE_16  0x20
#define TIMER_PRESCALE_32  0x28
#define TIMER_PRESCALE_64  0x30
#define TIMER_PRESCALE_128 0x38
```

```
// Timer prescale values
#define TIMER_PRESCALE_1   0x00
#define TIMER_PRESCALE_2   0x01
#define TIMER_PRESCALE_4   0x02
#define TIMER_PRESCALE_8   0x03
#define TIMER_PRESCALE_16  0x04
#define TIMER_PRESCALE_32  0x05
#define TIMER_PRESCALE_64  0x06
#define TIMER_PRESCALE_128 0x07
```

Slightly Higher level
SFR <u>bit</u> programming
#define and enums to make
more <u>readable code</u>

```c
#define uenum unsigned char enum
#define UCHAR unsigned char

uenum PRES_T {
       PRES_1 = 0,
       PRES_2 = 1,
       PRES_4 = 2,
       PRES_8 = 3,
       PRES_16 = 4,
       PRES_32 = 5,
       PRES_64 = 6,
       PRES_128 = 7
};

uenum TMODE_T {
       TMODE_ONE_SHOT = 0,
       TMODE_CONTINUOUS = 1,
       TMODE_COUNTER = 2,
       TMODE_PWM = 3,
       TMODE_CAPTURE = 4,
       TMODE_COMPARE = 5,
       TMODE_GATED = 6,
       TMODE_CAPTURE_COMPARE = 7
};

typedef struct {
    UCHAR TEN : 1;
    UCHAR TPOL : 1;
    UCHAR PRES : 3;
    uenum TMODE_T TMODE : 3;
} TxCTL1_T;
```

```c
#define T0CTL1_ (*((TxCTL1_T near*)&T0CTL1))
#define T1CTL1_ (*((TxCTL1_T near*)&T1CTL1))
#define T2CTL1_ (*((TxCTL1_T near*)&T2CTL1))
#define T3CTL1_ (*((TxCTL1_T near*)&T3CTL1))


typedef struct {
    UCHAR T2I : 1;
    UCHAR T1I : 1;
    UCHAR T0I : 1;
    UCHAR U0RXI : 1;
    UCHAR U0TXI : 1;
    UCHAR I2CI : 1;
    UCHAR SPII : 1;
    UCHAR ADCI : 1;
} IRQ0_T;

#define IRQ0_    (*((IRQ0_T near*)&IRQ0))
#define IRQ0ENH_ (*((IRQ0_T near*)&IRQ0ENH))
#define IRQ0ENL_ (*((IRQ0_T near*)&IRQ0ENL))
```

```
void init_timer_3(void) {

    T0CTL1_.TEN = 0;

    T0CTL1_.TMODE = TMODE_CONTINUOUS;
    T0CTL1_.PRES = PRES_128;
    T0CTL1_.TPOL = 0;

    T0CTL0 = 0;

    T0HL = 0x00;

    // reload = clock / prescale * timeout
    // Timeout = 0.05

    T0R = CLOCK / 128 * 0.05;

       // Enable Timer0 interrupt
    IRQ0ENL_.T0I = 1;

       // Configure the Output pin
       PADD &= ~b00000010;
       PAAF |= b00000010;

       T0CTL1_.TEN = 1;

}
```

```
#define uenum unsigned char enum
#define UCHAR unsigned char


uenum PRES_T {
       PRES_1 = 0,
       PRES_2 = 1,
       PRES_4 = 2,
       PRES_8 = 3,
       PRES_16 = 4,
       PRES_32 = 5,
       PRES_64 = 6,
       PRES_128 = 7
};


typedef struct {
    UCHAR TEN : 1;
    UCHAR TPOL : 1;
    UCHAR PRES : 3;
    uenum TMODE_T TMODE : 3;
} TxCTL1_T;
```

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	TPOL	PRES			TMODE		
RESET	0	0	000			000		
R/W	R/W	R/W	R/W			R/W		
ADDR	FF-E307H, FF-E317H, FF-E327H							

```
#define T0CTL1_ (*((TxCTL1_T near*)&T0CTL1))
#define T1CTL1_ (*((TxCTL1_T near*)&T1CTL1))
#define T2CTL1_ (*((TxCTL1_T near*)&T2CTL1))
#define T3CTL1_ (*((TxCTL1_T near*)&T3CTL1))
```

Define a set of enumerated constants (the value is important)

A bit-field declaration

Define a new macro of the new type at the address of the original macro

Timer, Part 2
Next week